# TD #1 – APL2

*12 novembre 2019*
*Erwan Kerrien*

Rappel : si vous avez des questions sur ce TD ou sur le cours, n'hésitez pas à m'envoyer un mail à Erwan.Kerrien@inria.fr (je consulte plus rarement mon mail Erwan.Kerrien@univ-lorraine.fr).

## 1 Calculs de complexité

Pour chaque algorithme suivant, donnez la valeur finale de $S$ puis donnez la complexité en $O(.)$, en fonction de $N$. On rappelle les formules utiles :

$$\sum_{k=0}^{N-1} 1 = N, \qquad \sum_{k=0}^{N-1} k = \frac{N(N-1)}{2}, \qquad \sum_{k=0}^{N-1} k^2 = \frac{N(N-1)(2N-1)}{6}$$

(a)

**Entrée** **:** $N$ : Entier
**Sortie** **:** $S$ : Entier
**V. I.** **:** $i$ :Entier
**Début**
  $S \leftarrow 0$
  **Pour** $i$ **allant de** *0* **à** $N-1$ **faire**
    $S \leftarrow S+1$
  **FinPour**
**Fin**

(b)

**Entrée** **:** $N$ : Entier
**Sortie** **:** $S$ : Entier
**V. I.** **:** $i, j$ :Entier
**Début**
  $S \leftarrow 0$
  **Pour** $i$ **allant de** *0* **à** $N-1$ **faire**
    **Pour** $j$ **allant de** *0* **à** $N-1$ **faire**
      $S \leftarrow S+1$
    **FinPour**
  **FinPour**
**Fin**

(c)

**Entrée** **:** $N$ : Entier
**Sortie** **:** $S$ : Entier
**V. I.** **:** $i, j$ :Entier
**Début**
  $S \leftarrow 0$
  **Pour** $i$ **allant de** *0* **à** $N-1$ **faire**
    **Pour** $j$ **allant de** *0* **à** $N*N-1$ **faire**
      $S \leftarrow S+1$
    **FinPour**
  **FinPour**
**Fin**

(d)

**Entrée** : $N$ : Entier
**Sortie** : $S$ : Entier
**V. I.** : $i, j$ :Entier
**Début**

```
  S ← 0
  Pour i allant de 0 à N − 1 faire
    Pour j allant de 0 à i faire
      S ← S + 1
    FinPour
  FinPour
```

**Fin**

(e)

**Entrée** : $N$ : Entier
**Sortie** : $S$ : Entier
**V. I.** : $i, j, k$ :Entier
**Début**

```
  S ← 0
  Pour i allant de 0 à N − 1 faire
    Pour j allant de 0 à i − 1 faire
      Pour k allant de j à i − 1 faire
        S ← S + 1
      FinPour
    FinPour
  FinPour
```

**Fin**

## 2 Chaînes de caractères

En C, une chaîne de caractères est représentée par un tableau de caractères dont le dernier élément a pour code ASCII 0, que l'on représente par le caractère `'\0'`. Cette marque de fin par un caractère non imprimable permet de ne pas avoir à passer la taille de la chaîne de caractères en paramètre de fonction. Par exemple la chaîne de caractères pour le mot «tableau» est stockée comme le tableau `['t','a','b','l','e','a','u','\0']`. Aucune fonction de base ne donne donc la longueur d'un tableau en C. Les fonctions `strlen` et `strcmp` correspondent cependant à de vraies fonctions C disponibles dans la bibliothèque standard (qui nécessite une instruction `#include <string.h>` en début de code).

1. Écrivez une fonction `strlen` qui calcule la longueur d'une chaîne de caractères `S`. Donnez sa complexité en fonction de la longueur de la chaîne (que l'on notera $N$).
2. Écrivez une fonction `Cmp` qui compare deux chaînes de caractères `S1` et `S2` et renvoie `Vrai` si `S1=S2` et `Faux` sinon. Donnez sa complexité. Est-il intéressant de débuter par un test sur les longueurs des chaînes (`S1` est différent de `S2` si elles sont de longueurs différentes) ?
3. Écrivez une fonction `strcmp` qui compare deux chaînes de caractères `S1` et `S2` selon leur ordre alphabétique et renvoie : `1` si `S1` est alphabétiquement avant `S2`, `-1` dans le cas contraire et `0` si `S1=S2`. Donnez sa complexité.

## 3 Distances

On considère un tableau de réels $T$ de taille $n$ et on cherche à calculer le plus grand écart entre deux éléments de ce tableau, soit :

$$M = \max_{i,j}(T_i - T_j)$$

ainsi que la somme

$$S = \sum_{i,j} (T_i - T_j)^2$$

1. Écrivez une fonction qui calcule $M$. Quelle en est la complexité ?

2. Même question pour $S$

## 4 Polynômes

Un polynôme de degré $n$ est de la forme $P(X) = p_0 + p_1 X + p_2 X^2 + \ldots + p_{n-1} X^{n-1} + p_n X^n = \sum_{i=0}^{n} p_i X^i$ avec $p_n \neq 0$. On peut donc le représenter sous la forme d'un tableau `P` qui contient les $n+1$ valeurs $(p_i)_{0 \leq i \leq n}$, tel que `P[i]`$=p_i$ pour tout $i$ entre 0 et $n$.

(a) Écrivez une fonction simple `Poly` qui prend en entrée un tableau `P`, un entier `n`, ainsi qu'un réel `x` et qui renvoie la valeur $P(x)$ où le polynôme $P$ est de degré `n` et ses coefficients $(p_i)$ sont stockés dans le tableau `P` (de taille $n+1$ donc). Donnez la complexité de cet algorithme en fonction de $n$.

(b) L'algorithme de Horner permet de minimiser le nombre de calculs nécessaires en remarquant l'identité suivante :

$$P(X) = p_0 + p_1 X + p_2 X^2 + \ldots p_n X^n = p_0 + X\left(p_1 + X\left(p_2 + \ldots X\left(p_{n-1} + X p_n\right)\ldots\right)\right)$$

Réécrivez la fonction précédente qui calcule $P(x)$, en suivant la méthode de Horner. Donnez sa complexité.